Algoritmos Paralelos y Distribuidos para Cómputo de Altas Prestaciones. Fundamentos, Aplicaciones y Evaluación de rendimiento

Marcelo Naiouf[(1)], Armando De Giusti[(1)(2)], Laura De Giusti[(1)], Franco Chichizola[(1)], Victoria Sanz[(1)(2)], Fabiana Leibovich[(1)], Enzo Rucci[(1)], Silvana Gallo[(1)], Erica Montes de Oca[(1)]
[(1)] **Instituto de Investigación en Informática LIDI (III-LIDI) - Facultad de Informática – UNLP** [(2)] **CONICET**
{mnaiouf, degiusti, ldgiusti, francoch, vsanz, fleibovich, erucci, sgallo, emontesdeoca}@lidi.info.unlp.edu.ar,

CONTEXTO

Se presenta una línea de Investigación que es parte del Proyecto 11/F011 "Procesamiento paralelo y distribuido. Fundamentos y aplicaciones en Sistemas Inteligentes y Tratamiento de imágenes y video" del III-LIDI acreditado por el Ministerio de Educación, y de proyectos acreditados y subsidiados por la Facultad de Informática de la UNLP.

Existe cooperación con Universidades de Argentina, Latinoamérica y Europa a través de proyectos acreditados por AECID, CyTeD, y OEI, y becas de Telefónica de Argentina. Asimismo, el Instituto forma parte del Sistema Nacional de Cómputo de Alto Desempeño (SNCAD).

RESUMEN

El eje central de la línea de investigación lo constituye el estudio de los temas de procesamiento paralelo y distribuido para cómputo de altas prestaciones, en lo referente a los fundamentos y a las aplicaciones. Incluye problemas de software asociados con la construcción, evaluación y optimización de algoritmos concurrentes, paralelos y distribuidos sobre arquitecturas multiprocesador.

Los temas de interés abarcan aspectos de fundamentos como diseño y desarrollo de algoritmos paralelos en diferentes arquitecturas multiprocesador y plataformas de software, paradigmas paralelos, modelos de representación de aplicaciones, mapeo o asignación de procesos a procesadores, métricas, escalabilidad, balance de carga, evaluación de performance. Las arquitecturas pueden ser homogéneas o heterogéneas.

Las áreas de experimentación se enfocan principalmente a la concepción de aplicaciones paralelas numéricas y no numéricas sobre grandes volúmenes de datos y/o de cómputo intensivo, con el fin de obtener soluciones de alto rendimiento.

Se han incorporado temáticas como el uso de GPU en el desarrollo de soluciones, y el análisis del consumo y la eficiencia energética en algoritmos paralelos.

Este proyecto se coordina con otros dos en curso en el III-LIDI, relacionados con Arquitecturas Distribuidas y Paralelas y Sistemas de Software Distribuido. Existe colaboración con el grupo CAOS del Departamento de Arquitectura de Computadores y Sistemas Operativos de la Univ. Autónoma de Barcelona en la dirección de tesis de postgrado.

Palabras clave: *Sistemas paralelos. Algoritmos paralelos y distribuidos. Clusters. Multicore. GPU. Balance de carga. Evaluación de performance. Consumo energético*

1. INTRODUCCION

Debido al interés por el desarrollo de soluciones a problemas con creciente demanda computacional y de almacenamiento, el procesamiento paralelo y distribuido se ha convertido en un área clave dentro de la Ciencia de la Computación, produciendo transformaciones en las líneas de I/D [BEN06] [BIS08][RAU10][SHA08][BEC08].

Más allá de las mejoras en la evolución de las arquitecturas físicas, el desafío está centrado en cómo aprovechar sus prestaciones. En este sentido, interesa realizar I/D en la especificación, transformación, optimización y evaluación de algoritmos distribuidos y paralelos, En esta línea de I/D la mayor importancia está en los algoritmos paralelos y en los métodos utilizados para su construcción y análisis [QIU08].

Uno de los cambios de mayor impacto ha sido el uso de manera masiva de procesadores con más de un núcleo (multinúcleo o multicore). Esto ha producido plataformas distribuidas híbridas (memoria compartida y distribuida), llevando a la necesidad de desarrollar sistemas operativos, lenguajes y algoritmos que las usen adecuadamente. También ha surgido la utilización de arquitecturas many-core como las placas gráficas de uso general como máquinas paralelas de memoria compartida, lo que constituye una plataforma con un paradigma de programación propio asociado.

La creación de algoritmos paralelos en arquitecturas multiprocesador, o la paralelización de un algoritmo secuencial, no es un proceso directo. El costo puede ser alto en términos del esfuerzo de programación [SHA08], y el manejo de la concurrencia adquiere un rol central en el desarrollo. Los pasos básicos para diseñar aplicaciones paralelas incluyen particionamiento, comunicación, aglomeración y mapeo de procesos a procesadores. Si bien en las primeras etapas el diseñador puede abstraerse de la máquina sobre la cual ejecutará el algoritmo, para obtener buen rendimiento en general debe tenerse en cuenta la plataforma de destino, dando lugar al concepto de *sistema paralelo* como combinación de hardware y software.

En las máquinas multiprocesador, se deben identificar las capacidades de procesamiento, interconexión, sincronización y escalabilidad [PAR09]. La caracterización y estudio de rendimiento del sistema de comunicaciones es de interés para la predicción y optimización de performance, así como la homogeneidad o heterogeneidad de los procesadores.

Muchos problemas algorítmicos se vieron fuertemente impactados por el surgimiento de las máquinas multicore (que integran dos o más núcleos computacionales dentro de un mismo chip), y la

tendencia creciente al uso de clusters de multicores. A partir de incorporar varios chips multicore dentro de un nodo y conectar múltiples nodos vía red, se puede crear una arquitectura NUMA, de modo que los cores en un chip compartan memoria principal, y puedan acceder remotamente a la memoria dedicada de otro chip, aunque ese acceso sea más costoso. Surgen varios niveles de comunicación: Intra CMP (2 cores del mismo chip), Inter CMP (2 cores que radican en distintos chips pero en el mismo nodo), e Inter Nodo (2 cores de 2 nodos distintos).

Esto impacta sobre el desarrollo de algoritmos que aprovechen adecuadamente las arquitecturas, y motiva el estudio de performance en sistemas híbridos [CHA07][SID07]. Además, es necesario estudiar la utilización de diferentes lenguajes ya que aún no se cuenta con un standard, aunque puede mencionarse el uso de MPI, OpenMP y Pthreads.

Para algunos problemas ha crecido la utilización de arquitecturas many-core como las placas gráficas de uso general (GPGPU, o *general purpose graphic processing unit*) como máquinas paralelas de memoria compartida [PIC11][KIR10]. Esto se debe a la gran cantidad de núcleos de procesamiento disponibles, buena performance y costo accesible. Dentro de los lenguajes asociados pueden mencionarse CUDA y OpenCL [LUE08][NOT09][NVI08].

Métricas de evaluación del rendimiento y balance de carga

La diversidad de opciones vuelve complejo el análisis de performance de los Sistemas Paralelos, ya que los ejes sobre los cuales pueden compararse dos sistemas son varios. Existe un gran número de métricas para evaluar el rendimiento, siendo las tradicionales tiempo de ejecución, speedup, eficiencia.

La *escalabilidad* permite capturar características de un algoritmo paralelo y la arquitectura en que se lo implementa. Posibilita testear la performance de un programa sobre pocos procesadores y predecirla en un número mayor, así como caracterizar la cantidad de paralelismo inherente en un algoritmo.

En arquitecturas distribuidas, los problemas que caracterizan el análisis de los algoritmos paralelos aparecen potenciados por las dificultades propias de la interconexión en una red en general no dedicada. Esto se torna más complejo aún si cada nodo puede ser un multicore con varios niveles de memoria.

El uso de procesadores con múltiples núcleos conlleva cambios en la forma de desarrollar aplicaciones y software, y evaluar su rendimiento. La cantidad de threads disponibles en estos sistemas también es importante, ya que su creación y administración requiere del uso de recursos como memoria; además los threads deben ser cuidadosamente planificados (scheduling) e incorporados en la pila de ejecución. En este sentido, el desarrollo de técnicas de scheduling eficientes es un tema de interés.

El objetivo primario del cómputo paralelo es reducir el tiempo de ejecución haciendo uso eficiente de los recursos. El *balance de carga* es un aspecto central y consiste en, dado un conjunto de tareas que comprenden un algoritmo y un conjunto de procesadores, encontrar el mapeo (asignación) de tareas a procesadores tal que cada una tenga una cantidad de trabajo que demande aproximadamente el mismo tiempo. Esto es más complejo si los procesadores (y comunicaciones) son heterogéneos. Dado que el problema general de mapping es *NP*-completo, pueden usarse enfoques que brindan soluciones subóptimas aceptables [OLI08]. Las técnicas de planificación a nivel micro (dentro de cada procesador) y macro (en un cluster) deben ser capaces de obtener buen balance de carga. Existen técnicas estáticas y dinámicas cuyo uso depende del conocimiento que se tenga sobre las tareas que componen la aplicación [LIU07][DUM08].

Evaluación de performance. Aplicaciones

Es importante referirse a un algoritmo paralelo mencionando el modelo de computación para el que fue diseñado. Uno de los objetivos en la definición del modelo es la posibilidad de predicción de performance que brinde el mismo, teniendo en cuenta conceptos tales como comunicación, sincronización y arquitectura física. El desarrollo de nuevos modelos requiere caracterizar el contexto de comunicaciones entre los procesadores y la asociación entre los algoritmos, el paradigma de cómputo paralelo elegido y la arquitectura de soporte.

En la evaluación de performance de distintas clases de aplicaciones sobre las arquitecturas, interesa estudiar la influencia de las estrategias de distribución de procesos y datos, y la carga (estática o dinámica) asignada a cada procesador sobre el speedup, eficiencia y escalabilidad. Un aspecto de interés que se ha sumado como métrica es el del consumo energético requerido [FEN05].

Entre las aplicaciones de interés se encuentran las numéricas y no numéricas con alta demanda de cómputo.

2. LINEAS DE INVESTIGACION y DESARROLLO

- Paralelización de algoritmos secuenciales. Diseño y optimización de algoritmos.
- Comparación de lenguajes y bibliotecas para procesamiento paralelo y distribuido.
- Arquitecturas multicore y many-core. Multithreading en multicore. Multiprocesadores distribuidos.
- Estudio de complejidad de algoritmos paralelos, en particular considerando multicore y heterogeneidad.
- Modelos y paradigmas de computación paralela.
- Programación sobre modelo híbrido (pasaje de mensajes y memoria compartida) en cluster de multicores.
- Modelos de representación y predicción de performance de algoritmos paralelos.
- Mapping y scheduling de aplicaciones paralelas sobre distintas arquitecturas multiprocesador.

- Análisis de los problemas de migración y asignación de procesos y datos a procesadores.
- Balance de carga estático y dinámico. Técnicas.
- Evaluación de performance. Speedup, eficiencia, escalabilidad.
- Análisis de consumo y eficiencia energética en algoritmos paralelos.
- Implementación de soluciones sobre diferentes modelos de arquitecturas homogéneas y heterogéneas.

3. RESULTADOS OBTENIDOS/ESPERADOS

- Formar RRHH en los temas del Subproyecto, incluyendo tesis de postgrado y tesinas de grado.
- Desarrollar y optimizar algoritmos paralelos sobre diferentes modelos de arquitectura. En particular, en aplicaciones numéricas y no numéricas de cómputo intensivo y tratamiento de grandes volúmenes de datos
- Utilizar arquitecturas híbridas que combinan memoria compartida y pasaje de mensajes, evaluando performance para distintos modelos de comunicación.
- Estudiar y desarrollar modelos de representación de aplicaciones paralelas y distribuidas y los algoritmos de mapeo (estático y dinámico).
- Desarrollar algoritmos paralelos sobre GPU. Para problemas regulares y con alta demanda de cómputo, comparar los resultados con otras plataformas.
- Evaluar la performance (eficiencia, rendimiento, speedup, escalabilidad) de las soluciones propuestas.
- Mejorar y adecuar las técnicas disponibles para el balance de carga (estático y dinámico).
- Estudiar y proponer las adecuaciones necesarias para los modelos de predicción y evaluación de performance con diferentes paradigmas de interacción entre procesos, y distintas arquitecturas de soporte.
- Estudiar y comparar los lenguajes sobre las plataformas mencionados.
- Adicionalmente, estudiar el impacto producido por los modelos de programación, lenguajes y algoritmos sobre el consumo y la eficiencia energética.
- Iniciar la investigación sobre la paralelización de aplicaciones en plataformas que combinan multicore y GPU, o que disponen de más de una GPU en una misma máquina.

En este marco, pueden mencionarse los siguientes resultados:

- Para la experimentación se utilizaron y analizaron diferentes arquitecturas homogéneas o heterogéneas, incluyendo multicores, cluster de multicores con 128 núcleos y GPU.

- Se experimentó la paralelización en arquitecturas híbridas, con el objetivo de estudiar el impacto del mapeo de datos y procesos, así como de los lenguajes y librerías utilizadas.

- Respecto de los aplicaciones estudiadas y algoritmos implementados, se trabajó fundamentalmente con los siguientes problemas:

  - **Best-first search paralelo sobre multicore y cluster de multicore:** El algoritmo de búsqueda A* (variante de Best-First Search) es utilizado como base para resolver problemas combinatorios y de planificación, donde se requiere encontrar una secuencia de acciones que minimicen una función objetivo para transformar una configuración inicial (problema a resolver) en una configuración final (solución). El alto requerimiento de memoria y cómputo causados por el crecimiento exponencial o factorial del grafo generado dinámicamente hacen imprescindible su paralelización, que permite beneficiarse de: (a) la gran cantidad de RAM y potencia de cómputo que provee un *cluster*, (b) la potencia de cómputo que proveen los procesadores *multicore*, (c) ambas características en caso de *cluster de muticore*. Tomando como caso de estudio el problema del N-Puzzle, se implementó el A* secuencial para resolverlo, y dos versiones del paralelo Hash Distributed A* (HDA*[KIS12] [BUR10]) que realiza el balance de carga de los nodos generados del grafo mediante una función de Hash: (1) una versión utiliza MPI, haciendo posible su ejecución tanto sobre arquitecturas con memoria distribuida y memoria compartida, y (2) otra versión utilizando Pthreads, para su ejecución sobre un multiprocesador con memoria compartida, que elimina ciertas ineficiencias respecto a (1) cuando la ejecución se realiza sobre memoria compartida, como son las replicaciones de datos entre procesos de estructuras comunes utilizadas y la serialización de datos para la comunicación de nodos entre procesos. Interesa realizar un análisis comparativo del rendimiento alcanzado por HDA*-MPI y HDA*-Pthreads sobre un multicore y el desarrollo de una versión de HDA* híbrida (Pthreads + MPI) para obtener mayor eficiencia al utilizar un cluster de multicore con respecto a la versión HDA*-MPI.

  - ***Diagonalización de matrices por el método de Jacobi sobre arquitecturas multicore***. El método de Jacobi para diagonalizar matrices simétricas tiene aplicaciones en áreas como biometría, visión artificial, procesamiento digital de señales, entre otros. El incremento en el volumen de datos de entrada provoca un aumento significativo en el tiempo de cómputo. La combinación de librerías de álgebra lineal optimizadas para la arquitectura subyacente, junto con la potencia que brinda un multicore y herramientas de programación paralela para dicha arquitectura permiten reducir el tiempo de ejecución. Se abordó el análisis del problema, y se estudiaron distintas implementaciones del algoritmo secuencial y optimizaciones posibles, la adaptación para usar una implementación de la API BLAS (Basic Linear Algebra Subprograms) optimizada para la arquitectura, y se implementó un algoritmo paralelo utilizando OpenMP. Se realizó un estudio de los tiempos de ejecución de las soluciones secuenciales, observando un mejor rendimiento para los algoritmos que hacen uso de librerías optimizadas para álgebra lineal (ATLAS). Se analizó el rendimiento (speedup, eficiencia) obtenido por el algoritmo paralelo propuesto sobre un multicore, a medida que se incrementa el volumen de datos de entrada (tamaño de la matriz) y al aumentar la cantidad de threads /cores, demostrando la escalabilidad del sistema paralelo

propuesto [SAN12]. Como líneas de trabajo futuro se plantea la migración del algoritmo paralelo para ejecutar sobre GPU, el estudio de la escalabilidad sobre dicha arquitectura, y el análisis del consumo energético de los algoritmos paralelos propuestos.

- **Aplicaciones con paralelismo de datos**. Se avanzó en la paralelización en cluster de multicores, tomando como caso de estudio una aplicación base en muchos problemas como la multiplicación de matrices. Se estudió la mejora introducida por las soluciones analizando escalabilidad en dos sentidos, al incrementar tanto el tamaño del problema como la cantidad de núcleos en la experimentación. Se implementaron diferentes soluciones híbridas (pasaje de mensajes y memoria compartida) con dos enfoques: comparación de librerías de programación paralela (MPI + Pthreads vs MPI + OpenMP), y comparación de performance alcanzable por algoritmos que aprovechan eficientemente el uso de la cache L1, de dos manera diferentes. Para este último análisis se implementó una solución que divide la matriz resultante en bloques para ser procesados al mismo tiempo que una segunda solución en la que las matrices de entrada son divididas en filas de bloques. Al aumentar el tamaño del problema, la segunda solución alcanza mejor performance dado que aprovecha la localidad espacial y temporal de la cache L1 más eficientemente [LEI12a][LEI12b]

- ***Análisis de Secuencias de ADN:*** el análisis de secuencias de ADN tiene múltiples aplicaciones, como la búsqueda de semejanzas entre dos de ellas. El gran tamaño que pueden alcanzar las secuencias (hasta 10^9 nucleótidos) y la complejidad computacional para compararlas por medio del algoritmo Smith-Waterman (orden N^2) hacen necesaria la paralelización [RUC11]..Se implementaron soluciones paralelas empleando un esquema de pipeline y con diferentes modelos de comunicación (memoria compartida, pasaje de mensajes y combinación de ambos). Los algoritmos se ejecutaron sobre un cluster de multicores homogéneo. Se realizó un análisis de los rendimientos obtenidos contemplando las métricas speedup y eficiencia [RUC12a]

- ***Construcción de árboles filogenéticos*:** filogenia es la relación entre los diferentes conjuntos de especies del planeta, la cual puede representarse mediante un árbol. Su tarea consiste en inferir dicho árbol a partir de las observaciones realizadas sobre los organismos existentes. Los avances en las tecnologías de secuenciación, que permiten obtener conjuntos de datos cada vez más grandes, y la complejidad computacional para construir los árboles por medio del método Neighbor-Joining (orden N^3) hacen necesaria su paralelización [STU88]. Se desarrollaron soluciones paralelas empleando un esquema maestro-esclavo y con diferentes modelos de comunicación (memoria compartida, pasaje de mensajes y combinación de ambos). Los algoritmos se ejecutaron sobre un cluster de multicores homogéneo, combinando memoria compartida y distribuida. Se estudió el desempeño individual de cada solución (teniendo en cuenta speedup y eficiencia) y se realizó un análisis comparativo de los mismos [RUC12b] [RUC13].

- ***Simulación de eventos discretos:*** los problemas de simulación implican sistemas con grandes necesidades de cómputo; en particular, se estudió la paralelización sobre cluster de multicores con tres modelos diferentes de comunicación: memoria compartida (utilizando OpenMP y Pthreads), memoria distribuida (usando MPI) y soluciones híbridas, y variando la distribución de datos y procesos [GAL12]. Actualmente se trabaja en la exploración de diferentes técnicas de particionado, mapping y balance de carga para tal arquitectura, y la evaluación de herramientas que incorporan dichas técnicas con el fin de obtener nuevas alternativas o mejoras en las mismas [SOL12].

- ***Problemas de tipo N-body.*** Se utilizaron las plataformas de memoria compartida GPU y cluster de multicore para la resolución de problemas con alta demanda computacional del tipo N-body. Se emplearon diferentes modelos de comunicación: memoria compartida (Pthread en CPU y CUDA en GPU), pasaje de mensajes (MPI) y soluciones híbridas (MPI-Pthread). Además, se analizó el problema desde el punto de vista energético, introduciendo los fundamentos de consumo energético. Se han mostrado los beneficios del uso de la GPU en problemas con características similares al caso planteado. Los tiempos de ejecución obtenidos son considerablemente inferiores comparados con las soluciones implementadas en CPU. Los experimentos realizados desde el punto de vista del consumo energético favorecen el uso de la GPU en tales problemas[KIR10] [MON12].

4. FORMACION DE RECURSOS HUMANOS

Dentro de la temática de la línea de I/D se concluyeron 3 Trabajos Finales de Especialización y 1 Tesina de Grado de Licenciatura. Se encuentran en curso en el marco del proyecto 8 tesis doctorales, 4 de maestría, 3 trabajos de Especialización y 3 Tesinas..

Se participa en el dictado de las carreras de Doctorado en Cs. Informáticas y Magíster y Especialización en Cómputo de Altas Prestaciones de la Facultad de Informática UNLP, por lo que potencialmente pueden generarse más Tesis y Trabajos Finales.

Existe cooperación con grupos de otras Universidades del país y del exterior, y hay tesistas de diferentes Universidades realizando su trabajo con el equipo del proyecto.

5. BIBLIOGRAFIA

[BEC08] Becker Alexander (Editor), “Concurrent and Parallel Computing: Theory, Implementation and Applications”, Nova Science Pub Inc, 2008, ISBN-10: 1604562749, ISBN-13: 9781604562743

[BEN06] Ben-Ari, M. “Principles of Concurrent and Distributed Programming,2/E”.Addison-Wesley. 2006

[BIS08] Bischof C., Bucker M., Gibbon P., Joubert G., Lippert T., Mohr B., Peters F. (eds.), Parallel Computing: Architectures, Algorithms and Applications, Advances in Parallel Computing, Vol. 15, IOS Press, February 2008.

[BUR10] Burns E, Lemons S, Ruml W, Zhou R. "Best First Heuristic Search for Multicore Machines". Journal of Artificial Intelligence Research, Vol.39, No.1, pp. 689-743, 2010.

[CHA07] Chapman B., The Multicore Programming Challenge, Advanced Parallel Processing Technologies; LNCS, Vol. 4847, p3, Springer, November 2007.

[DUM08] Dummler J., Ruaber T., Runger G., Mapping Algorithms for Multiprocessor Tasks on Multi-Core Clusters, Proc. 2008 International Conference on Parallel Processing IEEE CS 2008.

[FEN05] Feng, W.C., "The importance of being low power in HPC". Cyberinfrastructure Technology Watch Quarterly. 2005.

[GAL12] Gallo S., Chichizola F., De Giusti L., Naiouf M.. "Análisis de soluciones paralelas puras e híbridas en un problema de simulación". CACIC2012. ISBN: 978987-1648-34-4. Pág. 367-376. Octubre 2012.

[KIR10] Kirk D., Hwu W. "Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series). Morgan-Kaufmann. 2010.

[KIR10] Kirk, D.,Hwu, W. "Programming Massively Parallel Processors: A Hands-on Approach". ISBN: 978-0-12-381472-2. Elsevier, 2010.

[KIS12] Kishimoto A., Fukunaga A., Botea A. "Evaluation of a Simple, Scalable, Parallel Best-FirstSearch Strategy", Arxivpreprint: arXiv 1201.3204, 2012.

[LEI12a] Leibovich F., Chichizola F., De Giusti L., Naiouf M., Tirado, F., De Giusti, A. "Programación híbrida en clusters de multicore. Análisis del impacto de la jerarquía de memoria". CACIC2012. ISBN: 978987-1648-34-4. Pág. 306-315. 2012.

[LEI12b] Leibovich F., Naiouf M., De Giusti L., Tinetti F., De Giusti A.. "Hybrid Algorithms for Matrix Multiplication on Multicore Clusters". Procs PDPTA'12. Vol II. ISBN 1-60132-228-3J. 2012, USA.

[LUE08] Luebke D. "Cuda: Scalable parallel programming for high-performance scientific computing". 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2008.

[MON12] Montes de Oca E., De Giusti L., De Giusti A., Naiouf M. "Comparación del uso de GPU y cluster de multicore en problemas con alta demanda computacional". CACIC2012. ISBN: 978987-1648-34-4. Pág. 267-275. 2012.

[NOT09] Nottingham A. y Irwin B. "GPU packet classification using opencl: a consideration of viable classification methods". Research Conf. of the South African Inst. of Comp. Sc. and Inf. Technologists. ACM, 2009.

[NVI08] NVIDIA. "Nvidia CUDA compute unified device architecture, programming guide v.2.0". 2008.

[OLI08] Olivier S., Prins S., Scalable Dynamic Load Balancing Using UPC. Proc. 37th ICPP'08. CD-ROM, IEEE CS, September 2008.

[PAR09] Parashar M., Li Xiaolin, Chandra Sumir, "Advanced Computational Infrastructures for Parallel and Distributed Applications", Wiley-Interscience, 2009 ISBN-10: 0470072946

[PIC11] Piccoli M.F., "Computación de Alto Desempeño utilizando GPU". XV Escuela Internacional de Informática. Editorial Edulp, 2011.

[QIU08] Qiu X., Fox G. G., Yuan H., Bae S., Chrysanthakopoulos G., Nielsen H. F. "Performance of Multicore Systems on Parallel Data Clustering with Deterministic Annealing". LNCS 4331, pags. 407-416. ISBN 978-3-540-69383-3. Springer 2008.

[RAU10] Rauber T., Rünger G. "Parallel programming for multicore and cluster systems". Springer. 2010.

[RUC11] Rucci E., Chichizola F., De Giusti L., Naiouf M., De Giusti A. "DNA sequence alignment: hybrid parallel programming on a multicore cluster". Procs. ICDCC'11. España, 2011. ISBN 978-1-61804-030-5, pp 183-190.

[RUC12a] Rucci E., Chichizola F., Naiouf M., De Giusti L., De Giusti A. "Parallel pipelines for DNA sequence alignment on a cluster of multicores. A comparison of communication models". Journal of Communication and Computer. David Publishing Company, CA(EEUU) 2012, Vol9, N°12, pp. 1364-1371. ISSN: 1548-7709.

[RUC12b] Rucci E., Chichizola F., Naiouf M., De Giusti A. "Performance comparison of parallel programming paradigms on a multicore cluster". Procs PDCS 2012, Vol. 1, ACTA Press, Las Vegas, EEUU, ISBN: 978-0-88986-940-0, pp. 216-221.

[RUC13] Rucci E., Chichizola F., Naiouf M., De Giusti A. "A Hybrid Parallel Neighbor-Joining Algorithm for Phylogenetic Tree Reconstruction on a Multicore Cluster". To appear in Journal Parallel & Cloud Computing. American V-King Scientific Publishing, LTD, Nueva York (EEUU). ISSN print: 2304-9464. ISSN online: 2304-9456.

[SAN12] Sanz V., De Giusti A., Naiouf M. Performance Analysis of a Matrix Diagonalization Algorithm with Jacobi Method on a Multicore Architecture. Procs PDPTA 2012 Las Vegas, USA. ISBN: 1-60132-227-5. Vol I. pp. 883-889

[SHA08] Shavit N., Herlihy M., "The Art of Multiprocessor Programming", Morgan Kaufmann Pub, 2008, ISBN-13: 9780123705914

[SID07] Siddh S., Pallipadi V., Mallick A.. "Process Scheduling Challenges in the Era of Multicore Processors". Intel Technology Journal, Vol. 11, Issue 04, November 2007.

[SOL12] Solar Gallardo R., "Particionamiento y balance de carga en simulaciones distribuidas de bancos de peces", Tesis Doctoral, Universitat Autònoma de Barcelona, 2012

[STU88] J. Studier and K. Keppler, "A note on the neighbor-joining algorithm of Saitou and Nei," Mol. BioI.Evol., vol. 5, no. 6, 1988, pp. 729-731.

[TEL08] Teller J., Ozguner F., Ewing R., Scheduling Task Graphs on Heterogeneous Multiprocessors with Reconfigurable Hardware, Proc. 37th ICPP'08 IEEE CS, Sept. 2008